Orchestrating Orchestrators: Lessons learned and challenges faced while running Kubernetes at scale

Sidhartha Mani
@utter_babbage
github.com/wlan0
My journey so far...

Senior Software Engineer at Rancher Labs
Open Source Contributions

Docker Contributions

1. Syslog
Open Source Contributions

Docker Contributions

1. Syslog

`--log-driver=syslog`
Open Source Contributions

Docker Contributions

1. Syslog
   `--log-driver=syslog`
2. Logging flags
Open Source Contributions

Docker Contributions

1. Syslog
   `--log-driver=syslog`

2. Logging flags
   `--log-opt=max-size=1m`
Open Source Contributions

Docker Contributions

1. Syslog
   `--log-driver=syslog`
2. Logging flags
   `--log-opt=max-size=1m`
3. Log rotate
Open Source Contributions

Kubernetes Contributions

1. Cloud provider enhancements
Open Source Contributions

Kubernetes Contributions

1. Cloud provider enhancements
2. Affects API Server, Kubelet and Controller-Manager
Open Source Contributions

Kubernetes Contributions

1. Cloud provider enhancements
2. Affects API Server, Kubelet and Controller-Manager
3. New binary in Kubernetes (Cloud-Controller-Manager)
Open Source Contributions

Kubernetes Contributions

1. Cloud provider enhancements
2. Affects API Server, Kubelet and Controller-Manager
3. New binary in Kubernetes (Cloud-Controller-Manager)
4. More complexity in setup and operations
Kubernetes

Kubernetes is a **set of microservices** that work together to act as a **framework** for running distributed platforms.
Things we will cover

1. Setup of Kubernetes
   a. Setup
   b. Upgrades
2. Kubernetes Networking
   a. Choosing a provider
   b. Networking
3. Secrets and Config Management
Kubernetes

Etcd
API Server
Controller Manager
Scheduler
Proxy
Kubelet

Cluster Nodes
Kubernetes

- Etcd
- API Server
- Controller Manager
- Scheduler
- Proxy
- Kubelet

Storage

Orchestration

Node Agents

Cluster Nodes
Kubernetes

- Etdc
- API Server
- Controller Manager
- Scheduler
- Proxy
- Kubelet

Storage

Orchestration

Node Agents

Cluster Nodes
Kubernetes

- Etcld
- API Server
- Controller Manager
- Scheduler
- Proxy
- Kubelet

Cluster Nodes

- Storage
- Orchestration
- Node Agents
Run Kubernetes core, storage and workloads on separate machines to avoid setups with single points of failure.
Things we will cover

1. Setup of Kubernetes
   a. Setup
   b. Upgrades
2. Kubernetes Networking
   a. Networking
   b. Choosing a provider
3. Secrets and Config Management
Kubernetes Upgrade

Cluster Nodes

- Cloud Controller Manager
- Controller Manager
- API Server
- Etcd
- Scheduler
- Proxy
- Node Agents
- Kubelet
- Storage
- Orchestration
Always upgrade master before upgrading nodes, and etcd before master
Things we will cover

1. Setup of Kubernetes
   a. Setup
   b. Upgrades

2. Kubernetes Networking
   a. Choosing a provider
   b. Networking

3. Secrets and Config Management
Kubernetes Network Providers

1. Flannel
2. Project Calico
3. Weave Net
4. Contiv
5. GCE
6. OpenVSwitch
7. Open Virtual Networking
8. Romana
Kubernetes Network Providers

1. Flannel
2. Project Calico
3. Weave Net
4. Contiv
5. GCE
6. OpenVSwitch
7. Open Virtual Networking
8. Romana
# Kubernetes Network Providers

<table>
<thead>
<tr>
<th></th>
<th>Project Calico</th>
<th>Weave</th>
<th>Flannel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Isolation</td>
<td>Profile schema</td>
<td>CIDR schema</td>
<td>CIDR schema</td>
</tr>
<tr>
<td>Networking Model</td>
<td>Pure L3</td>
<td>VxLan or UDP</td>
<td>VxLan, UDP, host-gw...</td>
</tr>
<tr>
<td>Name Service</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Distributed Storage</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Encryption</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Protocol</td>
<td>TCP, UDP, ICMP, ICMP6</td>
<td>ALL</td>
<td>ALL</td>
</tr>
<tr>
<td>Partially connected</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Performance</td>
<td>Near Native</td>
<td>Near native in VxLan</td>
<td>Near native in VxLan</td>
</tr>
<tr>
<td>Feature</td>
<td>Project Calico</td>
<td>Weave</td>
<td>Flannel</td>
</tr>
<tr>
<td>------------------------------</td>
<td>-------------------------</td>
<td>--------------------------------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>Application Isolation</td>
<td>Profile schema</td>
<td>CIDR schema</td>
<td>CIDR schema</td>
</tr>
<tr>
<td>Networking Model</td>
<td>Pure L3</td>
<td>VxLan or UDP</td>
<td>VxLan, UDP, host-gw...</td>
</tr>
<tr>
<td>Name Service</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Distributed Storage</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Encryption</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Protocol</td>
<td>TCP, UDP, ICMP, ICMP6</td>
<td>ALL</td>
<td>ALL</td>
</tr>
<tr>
<td>Partially connected</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Performance</td>
<td>Near Native</td>
<td>Near native in VxLan</td>
<td>Near native in VxLan</td>
</tr>
</tbody>
</table>
# Kubernetes Network Providers

<table>
<thead>
<tr>
<th>Feature</th>
<th>Project Calico</th>
<th>Weave</th>
<th>Flannel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Isolation</td>
<td>Profile schema</td>
<td>CIDR schema</td>
<td>CIDR schema</td>
</tr>
<tr>
<td>Networking Model</td>
<td>Pure L3</td>
<td>VxLan or UDP</td>
<td>VxLan, UDP, host-gw...</td>
</tr>
<tr>
<td>Name Service</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Distributed Storage</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Encryption</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Protocol</td>
<td>TCP, UDP, ICMP, ICMP6</td>
<td>ALL</td>
<td>ALL</td>
</tr>
<tr>
<td>Partially connected</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Performance</td>
<td>Near Native</td>
<td>Near native in VxLan</td>
<td>Near native in VxLan</td>
</tr>
<tr>
<td></td>
<td>Project Calico</td>
<td>Weave</td>
<td>Flannel</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-------------------------</td>
<td>------------------------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>Application Isolation</td>
<td>Profile schema</td>
<td>CIDR schema</td>
<td>CIDR schema</td>
</tr>
<tr>
<td>Networking Model</td>
<td>Pure L3</td>
<td>VxLan or UDP</td>
<td>VxLan, UDP, host-gw...</td>
</tr>
<tr>
<td>Name Service</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Distributed Storage</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Encryption</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Protocol</td>
<td>TCP, UDP, ICMP, ICMP6</td>
<td>ALL</td>
<td>ALL</td>
</tr>
<tr>
<td>Partially connected</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Performance</td>
<td>Near Native</td>
<td>Near native in VxLan</td>
<td>Near native in VxLan</td>
</tr>
</tbody>
</table>
## Kubernetes Network Providers

<table>
<thead>
<tr>
<th></th>
<th>Project Calico</th>
<th>Weave</th>
<th>Flannel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Isolation</td>
<td>Profile schema</td>
<td>CIDR schema</td>
<td>CIDR schema</td>
</tr>
<tr>
<td>Networking Model</td>
<td>Pure L3</td>
<td>VxLan or UDP</td>
<td>VxLan, UDP, host-gw...</td>
</tr>
<tr>
<td>Name Service</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Distributed Storage</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Encryption</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Protocol</td>
<td>TCP, UDP, ICMP, ICMP6</td>
<td>ALL</td>
<td>ALL</td>
</tr>
<tr>
<td>Partially connected</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Performance</td>
<td>Near Native</td>
<td>Near native in VxLan</td>
<td>Near native in VxLan</td>
</tr>
</tbody>
</table>
Things we will cover

1. Setup of Kubernetes
   a. Setup
   b. Upgrades

2. Kubernetes Networking
   a. Choosing a provider
   b. Networking

3. Secrets and Config Management
Kubernetes Networking Model

1. Designed for portability from VMs
2. Every POD gets its own IP
Kubernetes Networking: Services

Logical set of pods

Pod 1: nginx
Pod 2: MySQL
Pod 3: Ghost
Kubernetes Networking Features

1. Selects Pods using labels
2. Best to create service before creating replication controllers
3. Create 1 replica first to check that it works and then scale it up

```yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
  labels:
    k8s-app: nginx-service
spec:
  ports:
  - port: 80
    targetPort: 80
    protocol: TCP
  name: http
selector:
  k8s-app: nginx-service
```
Kubernetes Networking

1. Microservices architecture
2. Built-in service discovery

```yaml
apiVersion: v1
type: Service
metadata:
  name: nginx-service
  labels:
    k8s-app: nginx-service
spec:
  ports:
  - port: 80
    targetPort: 80
    protocol: TCP
  name: http
  selector:
    k8s-app: nginx-service
```
Kubernetes Networking: Services

Pod 1: nginx
Pod 2: MySQL
Pod 3: Ghost

Pod 1: nginx
Pod 2: MySQL
Pod 3: Ghost

Pod 1: nginx
Pod 2: MySQL
Pod 3: Ghost
Kubernetes Networking

1. Microservices architecture
2. Built-in service discovery
3. Built-in load balancing
Kubernetes Networking: Services

Pod 1: nginx
Pod 2: MySQL
Pod 3: Ghost

Pod 1: nginx
Pod 2: MySQL
Pod 3: Ghost

Pod 1: nginx
Pod 2: MySQL
Pod 3: Ghost
Move ALL your legacy apps to microservice architecture in order to fully leverage Kubernetes
Do not use hostPort or nodePort services, unless absolutely necessary.
Kubernetes Networking

1. Microservices architecture
2. Built-in service discovery
3. Built-in load balancing
4. Ingress Support
   a. TLS terminations/passthrough
   b. SNI
   c. Wildcard based routing

```yaml
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: test
  annotations:
    http.port: "99"
spec:
  rules:
  - host: foo.bar.com
    http:
      paths:
      - path: /foo
        backend:
          serviceName: nginx-service
          servicePort: 80
```
Kubernetes Networking: Ingress

Pod 1: nginx
Pod 2: MySQL
Pod 3: Ghost

Pod 1: nginx
Pod 2: MySQL
Pod 3: Ghost
Kubernetes Networking

1. Microservices architecture
2. Built-in service discovery
3. Built-in load balancing
4. Ingress Support
   a. TLS terminations/passthrough
   b. SNI
   c. Wildcard based routing
5. Zero Downtime Upgrades
Kubernetes Networking

Pod 1: nginx
Pod 2: MySQL
Pod 3: Ghost

Pod 1: nginx
Pod 2: MySQL
Pod 3: Ghost
Pod 4: Ghostv2

Ingress
Kubernetes Networking

Ingress

Pod 1: nginx
Pod 2: MySQL
Pod 3: Ghost

Pod 4: Ghostv2

Pod 1: nginx
Pod 2: MySQL
Pod 3: Ghost
Pod 4: Ghostv2
Kubernetes Networking

```
kubectl apply -f service-update.yml
```
Define and use labels that identify semantic attributes of your application or resource.
Things we will cover

1. Setup of Kubernetes
   a. Setup
   b. Upgrades
2. Kubernetes Networking
   a. Choosing a provider
   b. Networking
3. Secrets and Config Management
Kubernetes Secrets

1. In-Built Secret Management
2. Encrypted over the network
3. Mounted directly into the containers

```
apiVersion: v1
type: Secret
metadata:
  name: mysecret
data:
  username: YWRtaW4=
  password: MWYyZDFIUmU2N2Rm
```
Kubernetes Secrets: Creation

1. From File

   kubectl create secret generic demo-secret --from-file=demo-secret.txt

2. Manually

   kubectl create -f demo-secret.yaml
Kubernetes Secrets: Usage

1. Mount Secret entirely into directory
2. Project only certain keys
3. Secrets as environment variables
4. Secret File Modes
5. Secret updates
Kubernetes Config Maps

1. In built Config Management
2. Similar to secrets but designed for non-sensitive information

apiVersion: v1
class: ConfigMap
metadata:
  name: special-config
  namespace: default
data:
  special.how: very
  special.type: charm
Kubernetes Config Maps: Creation

1. From File/Directory
   
   kubectl create configmap demo-config --from-file=demo-config.txt

2. Manually
   
   kubectl create -f demo-config.yaml

3. From Literal
   
   kubectl create config-map special-config --from-literal=special.how=very --from-literal=special.type=charm
Kubernetes Config Maps: Usage

1. Environment variables
2. Volume plugins
3. Command line arguments
Thank you

I will be at the booth on the Expo floor. (Look for Rancher)

@utter_babbage

Email: sid@rancher.com

Community Users: https://forums.rancher.com

Slack: https://rancher-users.slack.com

If you want to play with containers/microservices: https://try.rancher.com
Flannel

Configurable virtual overlay network

Requires that every machine should be able to talk to each other

Need to manage IP Address spaces

Need to manage another daemon - flanneld

Requires extra software to run - etcd

Need to setup and run etcd
Project Calico
Kubernetes Networking: DNS

Requirement from Kubernetes:

The IP address that a container sees itself as is the same IP address that others see it as
Kubernetes Networking: DNS

Requirement from Kubernetes:

The IP address that a container sees itself as is the same IP address that others see it as.